

Atty. Docket No. 166.0001  
Appl. No. 09/957,459

PATENT

**REMARKS/ARGUMENTS**

Claims 1, 36, and 44 have been amended to recite different language. Claim 5 has been amended to correct an antecedent basis problem. Claim 15 has been amended to improve readability. Claim 37 has been amended to correct a typographical error. New claims 58 and 59 have been added.

**1. Rejection of claims under 35 U.S.C. §103 over Koshisaka *et al.***

Claims 1-18 and 52-57 have been rejected under 35 U.S.C. §103(a) as being unpatentable over U.S. Patent No. 6,629,109, issued to Koshisaka *et al.* (hereinafter Koshisaka). See Office Action, at Page 2.

Applicants have amended independent claims 1 and 54 to include, "detecting an instruction ***by an operating system*** to perform an operation on an operating file." Applicants respectfully submit that a *prima facie* case of obviousness cannot be established for these claims. In particular, Koshisaka does not teach or suggest step 1 of newly amended independent claims 1 and 54, which includes, "detecting an instruction ***by an operating system*** to perform an operation on an operating file." See Office Action at Page 2. Filed herewith is a 37 CFR §1.132 Declaration of Steven Williams, an engineer in the field of software development. The facts set forth in the declaration of Mr. Williams together with the arguments set forth herein establish that there is no teaching or suggestion to modify Koshisaka to obtain Applicants' invention.

Koshisaka, the cited reference, teaches "an application-centric" file revision management system and method by which file revision management allegedly can be implemented even if applications are not provided with file revision management functions. Koshisaka teaches the use of an Applications Programming Interface (API) layer to be placed between an application and an operating system. According to Koshisaka, file backup reliability of the applications can be improved by this file revision management system. See Koshisaka, column 1, lines 57-63. See *a/so* Williams declaration, Paragraph 5. A file manipulation monitoring section in Koshisaka first detects file manipulation (e.g., file deletion or file name change) that is to be executed by the application by intercepting an instruction generated by the application. See Williams declaration, Paragraph 6. Koshisaka specifically states, "the file manipulation

Atty. Docket No. 166.0001  
Appl. No. 09/957,459

PATENT

monitoring section constantly monitors API (Application Program Interface) commands which are outputted by the application 1 to the operating system and thereby detects the file manipulation which is (going to be) executed by the application 1." See Koshisaka, column 6, lines 34-38.

The method of the present invention, as defined by newly amended claims 1 and 54 is a data-centric approach to file archiving, not an application-centric approach. This distinction is evidenced by the claim limitations of, "detecting an instruction *by an operating system* to perform an operation on an operating file," and "capturing the operating file temporally proximate to the operation being performed on the operating file, responsive to the detection of the instruction."

In direct contrast, Koshisaka is not data-centric; that is, it does not teach detection of an instruction by an operating system which is independent of application. Rather, the detected *instruction* or command in Koshisaka is *based on the API command* and is *by the application*, not *by the operating system*. See Williams declaration, Paragraphs 5, 6 and 7. For example, when an API command requesting file deletion is outputted by the application, the command is detected and hooked by the file manipulation monitoring section in the API. Subsequently, the processing section sends a different API command to the operating system. In other words, the instruction is first detected by the API and hooked. After the instruction is hooked, a different instruction is passed to the operating system, and the original command sent by the API to the operating system is not executed during performance of Koshisaka's revision management system activities. See Williams declaration, Paragraph 5.

As a result of detecting the instruction *by the application*, unlike the present invention, it is believed that Koshisaka provides a very limited layer of file protection, as file protection occurs only if the application is compatible with the Koshisaka assumptions of API behavior (Koshisaka, column 7, lines 59-64). See Williams declaration, Paragraph 8. Furthermore, the Examiner concedes that Koshisaka does not teach capturing the operating file and moving this captured file to an alternate storage location responsive to the detection of the instruction.

In view of the foregoing, the subject matter of claims 1 and 54 would not have been obvious in view of Koshisaka. It follows that claims 1 and 54 are allowable.

Atty. Docket No. 166.0001  
Appl. No. 09/957,459

PATENT

Claims 2-18 and 52-53 depend from independent claim 1. Thus, by definition, claims 2-18 and 52-53 are allowable over Koshisaka for at least the reasons offered with respect to claim 1. Claims 55-57 depend from claim 54. Thus, these claims are also allowable over Koshisaka for at least the reasons offered with respect to claim 54.

Regarding claim 4, on page 3 of the Office Action, the Examiner states that "Koshisaka teaches the archive files includes [sic] portions of the operating file." Column 6, lines 35-45 of Koshisaka, cited by the Office Action, discusses the file manipulation monitoring section of the file revision management system. In particular, the text discusses the file manipulation monitoring section's monitoring of API commands outputted by the application. Neither this section of Koshisaka nor any other section of the reference teaches or suggests creating an archive file "wherein the archive file includes portions of the operating file," as recited in claim 4. Applicants therefore submit that claim 4 is allowable over Koshisaka for this reason, in addition to the reason offered above with respect to claim 1 from which claim 4 depends.

Regarding claim 13, on page 4 of the Office Action, it is stated that Koshisaka teaches "determining whether the operating file has previously been captured prior to capturing the file." Column 6, lines 32-65, cited by the Office Action, discuss the file manipulation monitoring section. According to this text in Koshisaka, the file manipulation monitoring section instructs the processing section to store a deleted filename to a deleted file name memory section. The text does not teach or suggest any file capture, nor does it teach or suggest "determining whether the operating file has previously been captured prior to capturing the file." Applicants therefore respectfully submit that claim 13 is allowable over Koshisaka for this reason, in addition to the reason offered above with respect to claim 1 from which claim 13 depends.

**2. Rejection of claims under 35 U.S.C. §103 over Koshisaka *et al.* in view of Schmidt *et al.***

Claims 34-51 have been rejected under 35 U.S.C. §103(a) as allegedly being unpatentable over Koshisaka in view of U.S. Patent No. 6,535,894, issued to Schmidt *et al.* (hereinafter Schmidt). Applicants respectfully traverse this rejection. The Office

Atty. Docket No. 166.0001  
Appl. No. 09/957,459

PATENT

Action has failed to establish a *prima facie* case of obviousness and has fundamentally erred in assessing the scope and content of the references.

In particular, the Office Action has alleged that Koshisaka teaches, in a computing device, a method for archiving files including, detecting an instruction from a resident program to perform an operation on an operating file and creating an archive file from the operating file and storing the archive file in a first storage location temporally proximate to the operation being performed on the operating file, responsive to detecting the instruction. See Office Action, at Page 7. The Office Action states that Koshisaka does not explicitly teach the steps of "searching the first storage location for the archive file responsive to the occurrence of a first event" and "moving the archive file from the first storage location to the second storage location, responsive to a second event." The Office Action, however, alleges that Schmidt teaches these steps. See Office Action, at Page 7.

Schmidt, the cited reference, discloses an apparatus and method for incremental updating of archive files. According to Schmidt, an original archive file having one or more entries is created, where each entry in the original archive file is itself a file. The original archive file is transmitted to a client computer, and subsequently, a target archive file is created wherein one or more of its entries are "typically expected" to be identical to one or more entries in the original archive file according to Schmidt. A difference file including an index file describing the changes between the original archive file and the target archive file is also created and transmitted to the client computer. At the client computer, Schmidt teaches that the difference archive file is applied to the original archive file to produce a synthesized archive file.

Schmidt also describes a Java implementation which includes using HTML code to specify an archive file. See Schmidt, column 8, lines 2-11. According to this description, when a Java-enabled browser encounters the HTML code, it downloads an archive file from the server and searches the downloaded file for an archive file specified by a "Codebase" parameter. See Schmidt, column 8, lines 2-11.

The present invention, as defined by claim 34, includes the steps of "searching the first storage location for the archive file responsive to the *occurrence of a first event*" and "moving the archive file from the first storage location to the second storage

Atty. Docket No. 166.0001  
Appl. No. 09/957,459

PATENT

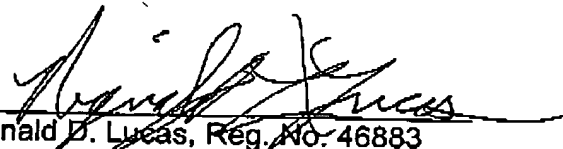
location **responsive to a second event.**" Thus, in the present invention, a first location is searched for an archive file in response to the occurrence of a first event, and the archive file is moved from the first location to a second location in response to a second event.

In direct contrast, in Schmidt's description of a Java implementation, upon encountering the HTML code, the "myarchive.jar" archive file is searched for and downloaded from the server. See Schmidt, column 8, lines 14-18. Thus, the searching and the moving (that is, the downloading of) of the myarchive.jar archive file both occur both in response to the same sole event, namely, encountering the HTML code. Unlike the present invention, Schmidt does not teach or suggest, "moving the archive file from the first storage location to the second storage location **responsive to a second event.**" Rather, there is only one event in Schmidt.

Therefore, claim 34 is allowable over Koshisaka in view of Schmidt as neither Koshisaka nor Schmidt, either alone or in combination, teaches or suggests the limitations of claim 34. Claims 35-51 depend from claim 34. Thus, by definition, claims 35-51 are allowable over Koshisaka in view of Schmidt for at least the reasons offered with respect to claim 34. Claims 55-57 depend from claim 54.

In view of the above Remarks, Applicants submit that all of the claims of the present invention are allowable and that the application is in condition for allowance. If the Examiner believes that the prosecution could be advanced through a telephone conversation, then the Examiner is invited to telephone the undersigned. Favorable action in this regard is earnestly solicited.

Respectfully submitted,  
CAHN & SAMUELS, L.L.P.

By:   
Reginald D. Lucas, Reg. No. 46883  
2000 P St., NW, Ste. 200  
Washington, D.C. 20036  
Telephone: (202) 331-8777  
Fax: (202) 331-3838

Atty. Docket No. 166.0001  
Appl. No. 09/957,459

PATENT

September 7, 2004